Chapitre III : Automate programmable

1. Introduction :

Les premiers automates programmables (Programmable Contrôler, PC) ont été introduits en 1969 aux Etats Unis pour remplacer les armoires à relais dans l'industrie de l'automobile. Depuis ils se sont répondus très rapidement dans le milieu industriel. Ce succès est dû en grande partie à leur faible coût et à la facilité avec laquelle ils peuvent être mis en œuvre.

Les automates programmables sont des micro-ordinateurs (machine électronique programmable) simplifiés qui sont spécialement conçus pour traiter par programme des problèmes de logique séquentielle, afin de remplacer les commandes d'automatismes en logique câblée.

- Avantage : utilisation de relais électromagnétiques et de systèmes pneumatiques pour la réalisation des parties commandes (Logique câblée).
- Inconvénients : cher, pas de flexibilité, pas de communication possible.
- Solution : utilisation de systèmes à base de microprocesseurs permettant une modification aisée des systèmes automatisés (Logique programmée).

L'objectif de ce chapitre est de présenter la structure interne et description des éléments d'un A.P.I, Choix de l'unité de traitement, Choix d'un automate programmable industriel, les interfaces d'entrées-sorties, les outils graphiques et textuels de programmation, la mise en œuvre d'un automate programmable industriel et les principes des réseaux d'automates.

2. Définition :

Un automate programmable est un appareil dédié au contrôle d'une machine ou d'un processus industriel, constitué de composants électroniques, comportant une mémoire programmable par un utilisateur non informaticien, à l'aide d'un langage adapté. En d'autres termes, un automate programmable est un calculateur logique, ou ordinateur, au jeu d'instructions volontairement réduit, destiné à la conduite et la surveillance en temps réel de processus industriels. Trois caractéristiques fondamentales distinguent totalement l'Automate Programmable Industriel (API) des ordinateurs (PC industriel ou autres) :

- il peut être directement connecté aux capteurs et pré-actionneurs grâce à ses entrées/sorties industrielles,
- il est conçu pour fonctionner dans des ambiances industrielles sévères (température, vibrations, micro-coupures de la tension d'alimentation, parasites, etc.),
- sa programmation à partir de langages spécialement développés pour le traitement de fonctions d'automatisme qui fait en sorte que sa mise en œuvre et son exploitation ne nécessitent aucune connaissance en informatique.

3. Structure d'un système automatisé :

Tout système automatisé peut se décomposer selon le schéma ci-dessous :

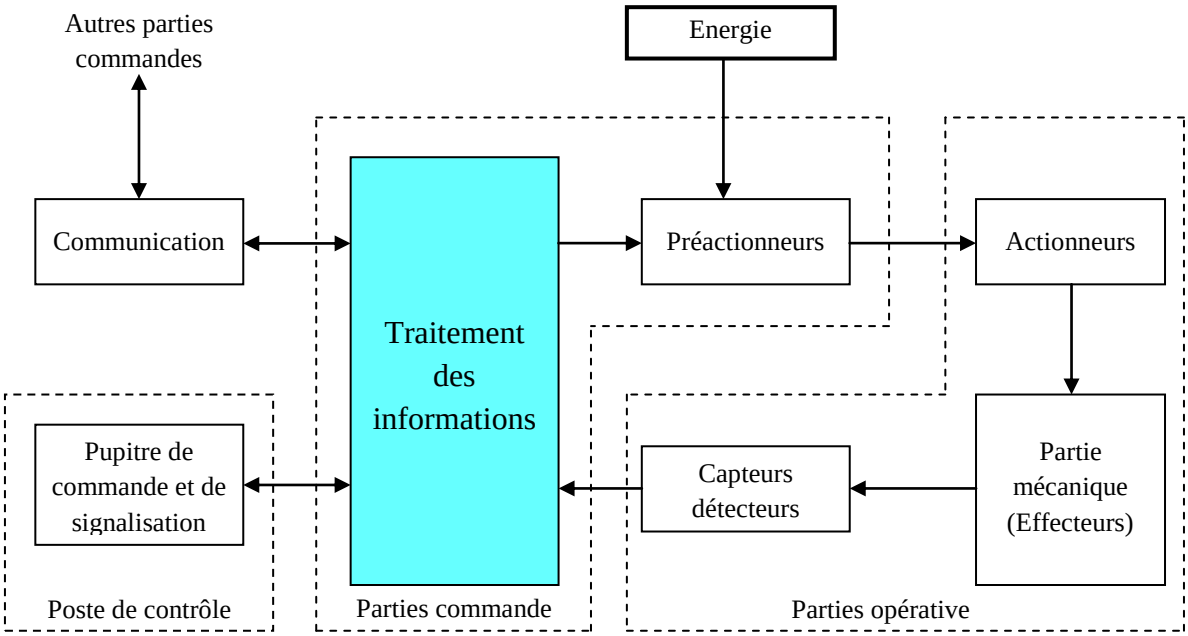


Figure (III.1) : Architecture d'un système automatisé.

3.1 Partie opérative :

Elle agit sur la matière d'œuvre afin de lui donner sa valeur ajoutée. Les actionneurs (moteurs, vérins) agissent sur la partie mécanique du système qui agit à son tour sur la matière d'œuvre. Les capteurs / détecteurs permettent d'acquérir les divers états du système.

3.2 Partie commande :

Elle donne les ordres de fonctionnement à la partie opérative. Les préactionneurs permettent de commander les actionneurs ; ils assurent le transfert d'énergie entre la source de puissance (réseau électrique, pneumatique …) et les actionneurs. Exemple : contacteur, distributeur …

Ces préactionneurs sont commandés à leur tour par le bloc traitement des informations. Celui-ci reçoit les consignes du pupitre de commande (opérateur) et les informations de la partie opérative transmises par les capteurs / détecteurs.

En fonction de ces consignes et de son programme de gestion des tâches (implanté dans un automate programmable ou réalisé par des relais (on parle de logique câblée)), elle va commander les préactionneurs et renvoyer des informations au pupitre de signalisation ou à d'autres systèmes de commande et/ou de supervision en utilisant un réseau et un protocole de communication.

3.3 Poste de contrôle :

Composé des pupitres de commande et de signalisation, il permet à l'opérateur de commander le système (marche, arrêt, départ cycle …). Il permet également de visualiser les

différents états du système à l'aide de voyants, de terminal de dialogue ou d'interface homme-machine (IHM).

3.3.1 Domaines d'emploi des automates :

On utilise les API dans tous les secteurs industriels pour la commande des machines (convoyage, emballage ...) ou des chaînes de production (automobile, agroalimentaire ...) ou il peut également assurer des fonctions de régulation de processus (métallurgie, chimie ...).

Il est de plus en plus utilisé dans le domaine du bâtiment (tertiaire et industriel) pour le contrôle du chauffage, de l'éclairage, de la sécurité ou des alarmes.

3.3.2 Nature des informations traitées par l'automate :

Les informations peuvent être de type :

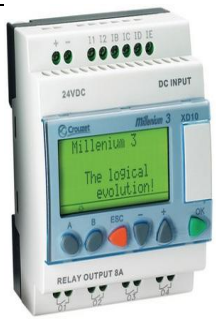
- Tout ou rien (T.O.R.) : l'information ne peut prendre que deux états (vrai/faux, 0 ou 1 …). C'est le type d'information délivrée par un détecteur, un bouton poussoir … f.
- Analogique : l'information est continue et peut prendre une valeur comprise dans une plage bien déterminée. L'information délivrée par un capteur (pression, température …).
- Numérique : l'information est contenue dans des mots codés sous forme binaire ou bien hexadécimale. C'est le type d'information délivrée par un ordinateur ou un module intelligent.

4. Architecture des automates :

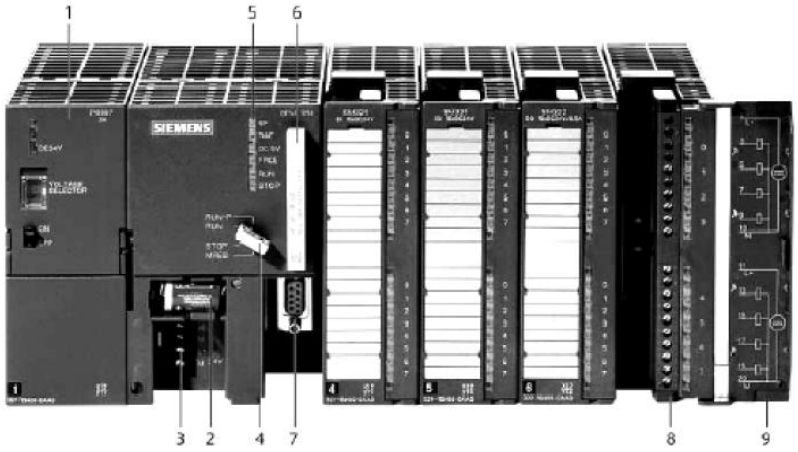
4.1 Aspect extérieur :

Les automates peuvent être de type compact ou modulaire.

– L'API de type compact : on distinguera les modules de programmation (LOGO de Siemens, ZELIO de Schneider, MILLENIUM de Crouzet ...) des microautomates. Il intègre le processeur, l'alimentation, les entrées et les sorties. Selon les modèles et les fabricants, il pourra réaliser certaines fonctions supplémentaires (comptage rapide, E/S analogiques ...) et recevoir des extensions en nombre limité. Ces automates, de fonctionnement simple, sont généralement destinés à la commande de petits automatismes.

SIEMENS LOGO	CROUZET MILLENIUM	SCHNEIDER ZELIO	SCHNEIDER TWIDO	MOELLER PS4
				

- De type modulaire, le processeur, l'alimentation et les interfaces d'entrées/sorties résident dans des unités séparées (modules) et sont fixées sur un ou plusieurs racks contenant le "fond de panier" (bus plus connecteurs). Ces automates sont intégrés dans les automatismes complexes où puissance, capacité de traitement et flexibilité sont nécessaires.



1-Module d'alimentation
2-Pile de sauvegarde
3-Connexion au 24V cc
4-Commutateur de mode (à clé)
5-LED de signalisation d'état et de défauts
6-Carte mémoire
7-Interface multipoint (MPI)
8-Connecteur frontal
9-Volet en face avant

Figure (III.2) : Automate modulaire (SIEMENS).

4.2 Structure interne :

La structure interne d'un API peut se représenter comme suit :

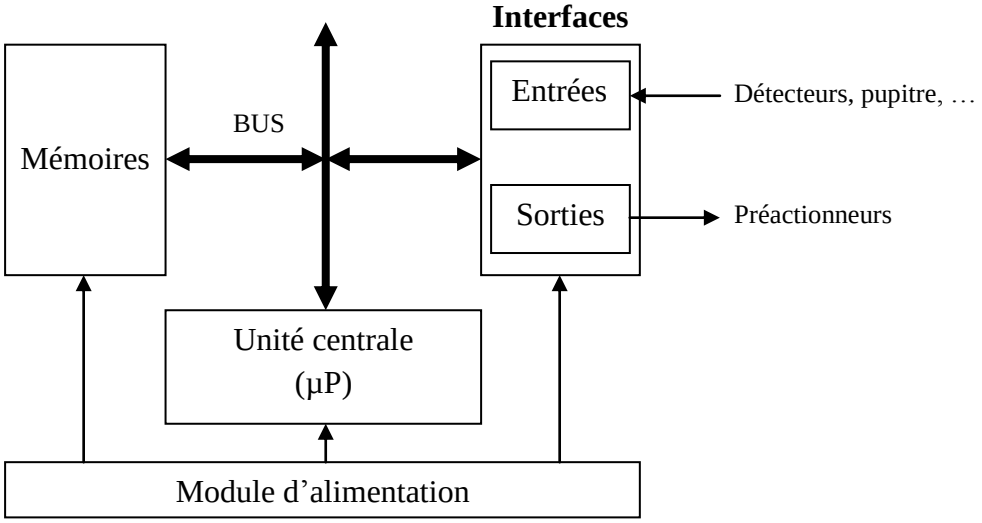


Figure (III.3) : Structure interne d'un API.

- Module d'alimentation : il assure la distribution d'énergie aux différents modules.
- Unité centrale : à base de microprocesseur, elle réalise toutes les fonctions logiques, arithmétiques et de traitement numérique (transfert, comptage, temporisation ...).

- Le bus interne : il permet la communication de l'ensemble des blocs de l'automate et des éventuelles extensions.
- Mémoires : Elles permettent de stocker le système d'exploitation (ROM ou PROM), le programme (EEPROM) et les données système lors du fonctionnement (RAM). Cette dernière est généralement secourue par pile ou batterie. On peut, en règle générale, augmenter la capacité mémoire par adjonction de barrettes mémoires type PCMCIA.
- Interfaces d'entrées/sorties :
  - Interface d'entrée : elle permet de recevoir les informations du S.A.P. ou du pupitre et de mettre en forme (filtrage, ...) ce signal tout en l'isolant électriquement (optocouplage).
  - Interface de sortie : elle permet de commander les divers préactionneurs et éléments de signalisation du S.A.P. tout en assurant l'isolement électrique.

Figure (III.4) : Architecture réelle d'un API S7-300.

4.3 Fonctions réalisées :

Les automates compacts permettent de commander des sorties en T.O.R et gèrent parfois des fonctions de comptage et de traitement analogique.

Les automates modulaires permettent de réaliser de nombreuses autres fonctions grâce à des modules intelligents que l'on dispose sur un ou plusieurs racks. Ces modules ont l'avantage de ne pas surcharger le travail de la CPU car ils disposent bien souvent de leur propre processeur.

4.4 Principales fonctions :

- Cartes d'entrées/sorties : Au nombre de 4, 8, 16 ou 32, elles peuvent aussi bien réaliser des fonctions d'entrées, de sorties ou les deux.
  - Ce sont les plus utilisées et les tensions disponibles sont normalisées (24, 48, 110 ou 230V continu ou alternatif ...).
  - Les voies peuvent être indépendantes ou posséder des "communs".
  - Les cartes d'entrées permettent de recueillir l'information des capteurs, boutons ... qui lui sont raccordés et de la matérialiser par un bit image de l'état du capteur.

- Les cartes de sorties offrent deux types de technologies : les sorties à relais électromagnétiques (bobine plus contact) et les sorties statiques (à base de transistors ou de triacs).

– Cartes de comptage rapide : elles permettent d'acquérir des informations de fréquences élevées incompatibles avec le temps de traitement de l'automate.

Exemple : signal issu d'un codeur de position.

– Cartes de commande d'axe : Elles permettent d'assurer le positionnement avec précision d'élément mécanique selon un ou plusieurs axes. La carte permet par exemple de piloter un servomoteur et de recevoir les informations de positionnement par un codeur. L'asservissement de position pouvant être réalisé en boucle fermée.

– Cartes d'entrées/sorties analogiques : Elles permettent de réaliser l'acquisition d'un signal analogique et sa conversion numérique (CAN) indispensable pour assurer un traitement par le microprocesseur.

- La fonction inverse (sortie analogique) est également réalisée.
- Les grandeurs analogiques sont normalisées : 0-10V ou 4-20mA.

– Autres cartes :

- Cartes de régulation PID
- Cartes de pesage
- Cartes de communication (Ethernet ...)
- Cartes d'entrées/sorties déportées

4.5 Choix d'un automate programmable industriel :

Le choix d'un API est adapté aux besoins après l'établissement du cahier des charges. On doit tenir compte de plusieurs critères à savoir :

– Le nombre et la nature d'entrées/sorties intégrés;
– La nature du traitement (temporisation, comptage, ...) ;
– Les moyens de dialogue et le langage de programmation;
– La communication avec les autres systèmes;
– Les moyens de sauvegarde du programme ;
– La fiabilité, robustesse et immunité aux parasites ;
– Capacité de la mémoire ;
– La documentation, le service après vente, durée de la garantie et la formation.

5. Câblage des entrées/sorties d'un automate :

L'automate est alimenté généralement par le réseau monophasé 230V/50Hz mais d'autres alimentations sont possibles (110V, …, etc.).

La protection sera de type magnéto-thermique (voir les caractéristiques de l'automate et les préconisations du constructeur).

Il est souhaitable d'asservir l'alimentation de l'automate par un circuit de commande spécifique (contacteur KM1). De même, les sorties seront asservies au circuit de commande et alimentées après validation du chien de garde.

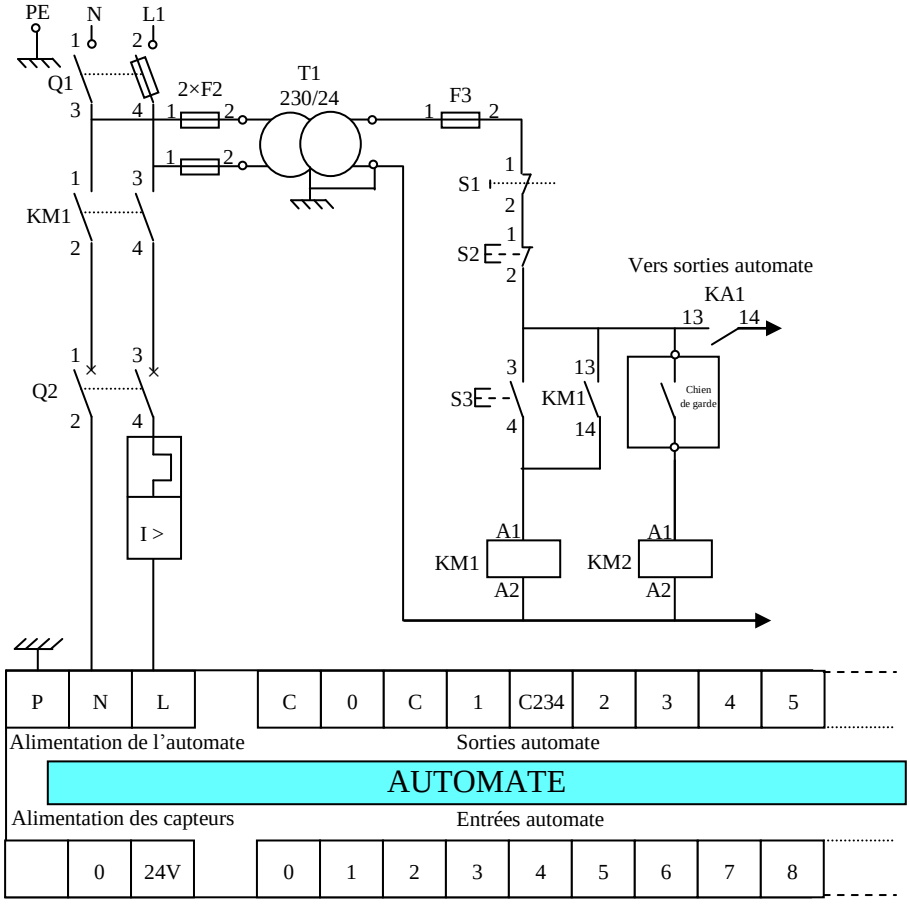


Figure (III.5) : Raccordement automate.

5.1 Alimentation des entrées de l'automate :

L'automate est pourvu généralement d'une alimentation pour les capteurs/détecteurs (attention au type de logique utilisée : logique positive ou négative).

Les entrées sont connectées au OV (commun) de cette alimentation. Les informations des capteurs/détecteurs sont traitées par les interfaces d'entrées.

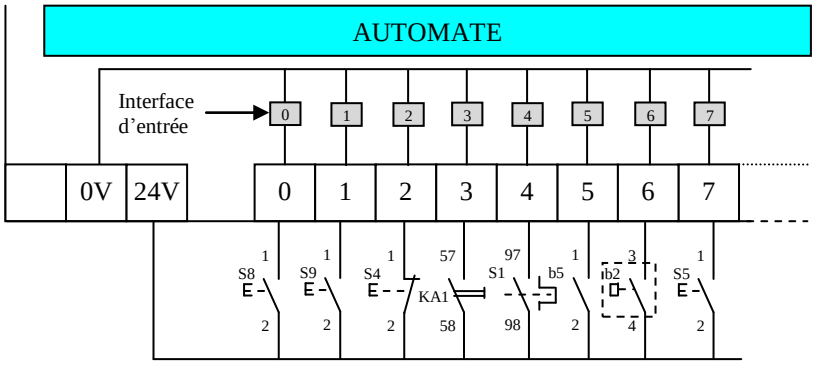


Figure (III.6) : Câblage des entrées de l'automate.

5.2 Alimentation des sorties de l'automate :

Les interfaces de sorties permettent d'alimenter les divers préactionneurs. Il est souhaitable d'équiper chaque préactionneur à base de relais de circuits RC (non représentés).

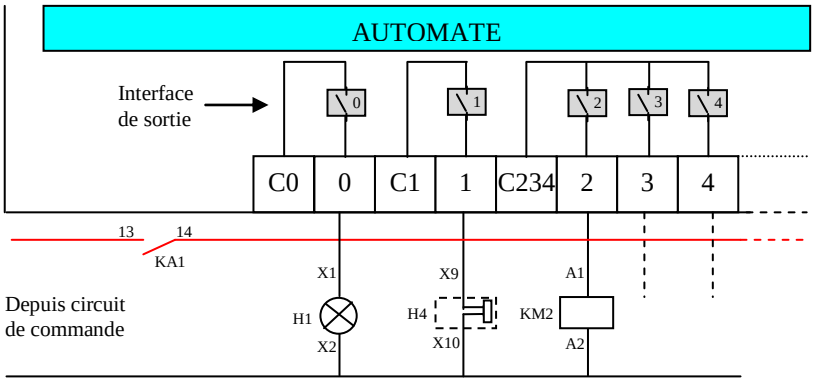
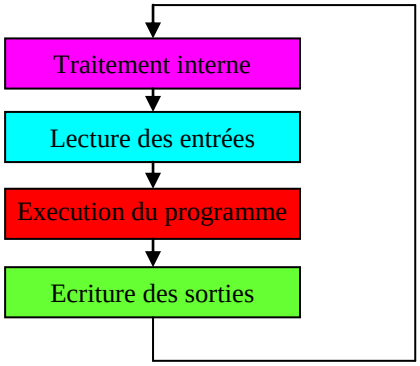


Figure (III.7) : Câblage des sorties de l'automate.

5.3 Traitement du programme automate :

Tous les automates fonctionnent selon le même mode opératoire :



- Traitement interne : L'automate effectue des opérations de contrôle et met à jour certains paramètres systèmes (détection des passages en RUN / STOP, mises à jour des valeurs de l'horodateur, ...).
- Lecture des entrées : L'automate lit les entrées (de façon synchrone) et les recopie dans la mémoire image des entrées.
- Exécution du programme : L'automate exécute le programme instruction par instruction et écrit les sorties dans la mémoire image des sorties.
- Ecriture des sorties : L'automate bascule les différentes sorties (de façon synchrone) aux positions définies dans la mémoire image des sorties.

Ces quatre opérations sont effectuées continuellement par l'automate (fonctionnement cyclique). On appelle scrutation l'ensemble des quatre opérations réalisées par l'automate et le temps de scrutation est le temps mis par l'automate pour traiter la même partie de programme. Ce temps est de l'ordre de la dizaine de millisecondes pour les applications standards. Le temps de réponse total (TRT) est le temps qui s'écoule entre le changement d'état d'une entrée et le changement d'état de la sortie correspondante.

6. Les langages de programmation :

Différentes méthodes permettent de réaliser un programme d'automatisme. Le choix dépend essentiellement des personnes qui sont amenées à le concevoir. Parmi ces méthodes :

6.1 La méthode booléenne :

Le programme se présente sous forme d'une suite d'équations logiques.

$S = (b + S/d)/a$ $R15(/AI2 + BI5)(EIO + E46) = A30$

6.2 La méthode a schémas logiques (les logigrammes) :

C'est déjà l'amorce du langage automate : Si absence de « a » (/a) et si présence de « b » ou si présence de « S » et absence de « d » (//d) alors la sortie « S » est validée.

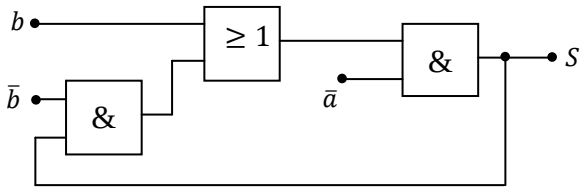


Figure (III.8) : Schémas logiques.

6.3 La méthode a relais :

Il s'agit dans ce cas, du schéma de câblage des électriciens. L'écriture d'un programme est par juxtaposition de symboles qui forment un schéma dont l'allure est telle qu'on parle de «schéma en échelle».

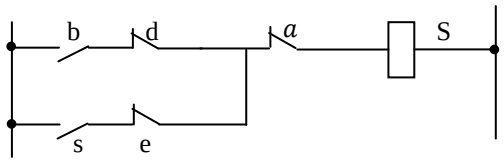


Figure (III.9) : Méthode a relais.

6.4 La méthode informatique (organigrammes) :

Le programme se présente sous forme d'organigramme.

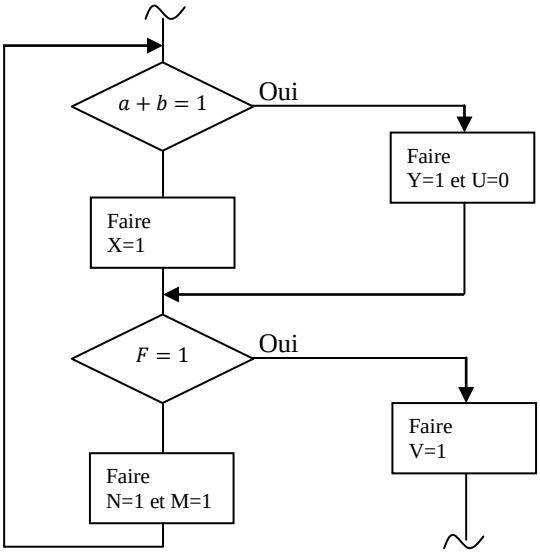


Figure (III.10) : Forme d'organigramme.

6.5 La méthode par étapes :

A l'aide de GRAFCET ou des réseaux de PETRI, cette méthode d'étude des systèmes séquentiels associe à chaque séquence une étape.

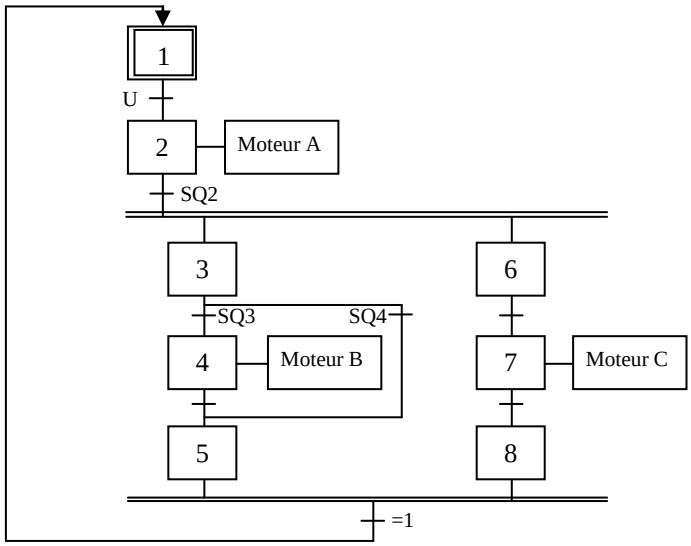


Figure (III.11) : Méthode grafcet.

6.6 Choix de langage :

Les fournisseurs d'automates programmables et les sociétés de services spécialisées en automatismes, proposent un large choix de langages de programmation. Ces programmes fonctionnent sur des plates-formes matérielles adaptées aux différentes applications. Dans les bureaux d'étude, les utilisateurs ont la possibilité de réaliser leurs programmes sur un PC. Pour les ateliers aux ambiances plus sévères, ont été conçues des consoles de programmation très fiables, adaptées aux besoins. Dans certains cas d'applications, cependant, de simples ordinateurs portables peuvent assurer ces taches avec succès.

L'un des principaux soucis de l'automaticien est de réduire le temps de développement du programme, pas étonnant donc que l'on utilise des sous-programmes destinés à traiter les fonctions répétitives. Siemens, premier à avoir adopté cette démarche avec son logiciel Step 5, "programmation structurée". Il s'agit d'un assemblage de modules fonctionnels (livrés par le fournisseur). La qualité du programme est garantie car les modules peuvent être testés peu à peu avant leur mise en œuvre. Sa simplicité entraîne un gain de temps. Aujourd'hui, tous les constructeurs proposent ce type de programmation.

7. Langage à contacts (LD : Lader diagram) :

Le langage LD (lader diagram) est une représentation graphique d'équations booléennes combinant des contacts (en entrée) et des relais (en sortie). Il permet la manipulation de données booléennes, à l'aide de symboles graphiques organisés dans un diagramme comme les éléments d'un schéma électrique à contacts. Les diagrammes LD sont limités à gauche et à droite par des barres d'alimentation. Il utilise les symboles tels que : contacts, relais, …etc.

7.1 Mise en équations des GRAFCET :

Malheureusement, ce ne sont pas tous les automates qui se programment en GRAFCET directement. Mais, généralement ils peuvent être programmés en langage à contacts (LD).

Il faut donc pouvoir transformer le GRAFCET qui est la meilleure approche qui existe pour traiter les systèmes séquentiels en « diagramme échelle » qui est le langage le plus utilisé par les automates.

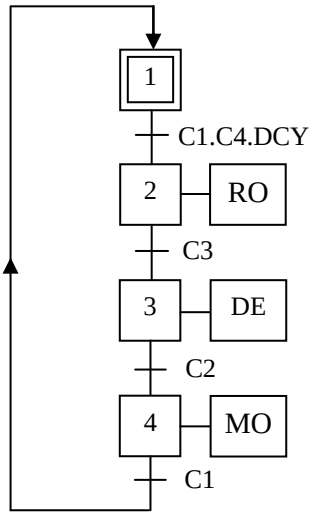
7.1.1 Mémoire d'étape :

Afin de respecter les règles d'évolution du GRAFCET, chaque étape peut être matérialisée par une mémoire du type marche prioritaire possédant une structure de la forme :

$X = Encl + \overline{RAZ}.X$

Les termes d'enclenchement et de remise à zéro sont définis de la manière suivante :

$$\text{ETAPE X} \begin{cases} \text{Encl: Etat logique de l'Etape(s)précédente(s). Réceptivité} \\ \text{RAZ: Etat logique de l'Etape(s) suivante(s)} \end{cases}$$



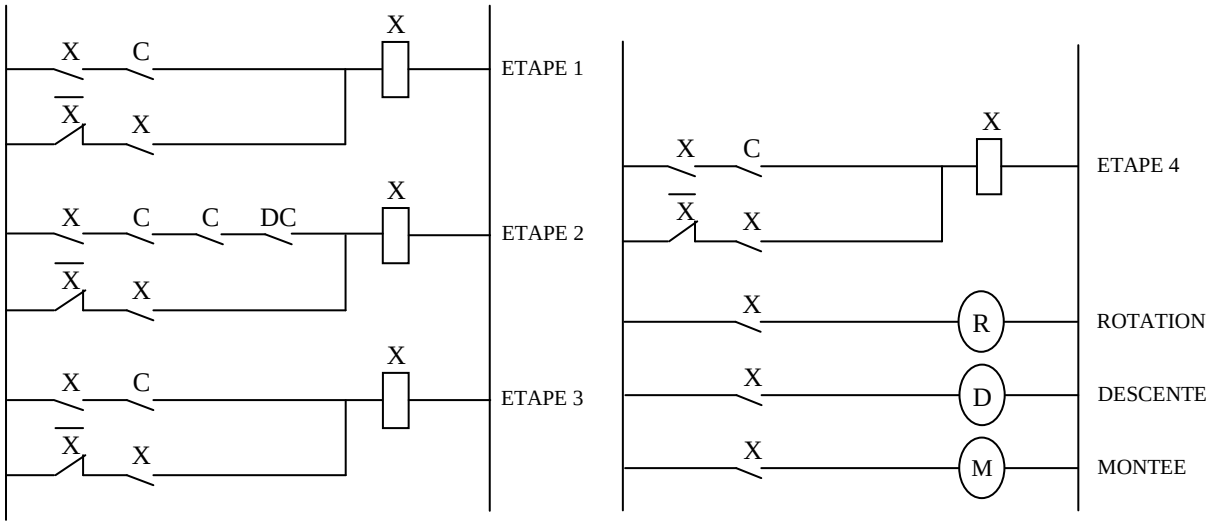
$$\text{X1} \begin{cases} \text{Encl: X4. C1} \\ \text{RAZ: X2} \end{cases} \qquad X1 = \text{X4. C1} + \overline{X2}.X1$$

$$\text{X2} \begin{cases} \text{Encl: X1. C1. C4. DCY} \\ \text{RAZ: X3} \end{cases} \qquad X2 = \text{X1. C1. C4. DCY} + \overline{X3}.X2$$

$$\text{X3} \begin{cases} \text{Encl: X2. C3} \\ \text{RAZ: X4} \end{cases} \qquad X3 = \text{X2. C3} + \overline{X4}.X3$$

$$\text{X4} \begin{cases} \text{Encl: X3. C2} \\ \text{RAZ: X1} \end{cases} \qquad X4 = \text{X3. C2} + \overline{X1}.X4$$

Les équations des mémoires étape déterminée précédemment nous donnent le schéma de câblage électrique suivant :



Pour établir la commande de chaque sortie, il suffit de considérer la ou les étapes durant lesquelles la sortie doit être enclenchée. Ainsi :

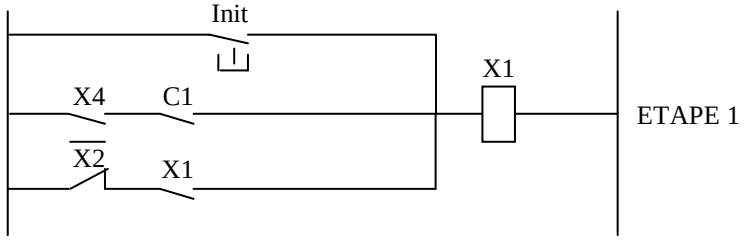
- La sortie RO a lieu durant l'ETAPE 2 d'où RO = X2
- La sortie DE a lieu durant l'ETAPE 3 d'où DE = X3
- La sortie MO a lieu durant l'ETAPE 4 d'où MO = X4

7.1.2 Initialisation de la séquence :

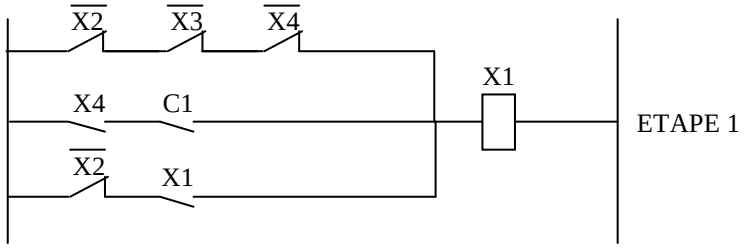
Nous remarquons sur le schéma précédent qu'à la mise sous tension, toutes les mémoires se trouvant ici à l'état repos, aucune évolution n'est possible.

Il est donc impératif d'initialiser la séquence en venant enclencher la mémoire X1 matérialisant l'étape initiale de notre GRAFCET. Ceci est obtenu :

- Soit en utilisant un contact d'initialisation ou un contact de passage commandé lors de la mise sous tension de l'automatisme, comme le montre le schéma suivant :



- Soit en testant l'état repos de toutes les mémoires d'étape suivantes, pour venir alors systématiquement enclencher la mémoire X1, comme le montre le schéma suivant :



7.2 Le langage LADER (LD) :

Les langages de programmation des API sont de natures diverses étant donné la diversité, des utilisateurs pouvant les utiliser.

Le langage LADER des API d'origine américaine utilise le symbolisme classique des schémas à relais accompagné de blocs graphiques préprogrammés pour réaliser des fonctions d'automatisme (calculs, temporisation, compteur,…..).

C'est une suite de réseaux qui seront parcourus séquentiellement. Les entrées sont représentées par des interrupteurs -| |- ou -|/|- si entrée inversée, les sorties par des bobines **-( )-** ou des bascules **-(S)- -(R)-**.Il y a également d'autre opérations :

- l'inverseur **-|NOT|-**,
- l'attente d'un front montant **-(P)-** ou descendant **-(N)-**.

Les sorties sont obligatoirement à droite du réseau On doit évidemment identifier nos **E/S**, soit directement par leur code **(Ia.b / Qa.b)**, ou avec leur libellé en clair défini dans la table des mnémoniques.

On relie les éléments en série pour la fonction **ET**, en parallèle pour le **OU**. On peut utiliser des bits internes (peuvent servir en bobines et interrupteurs), comme on utilise dans une calculatrice une mémoire pour stocker un résultat intermédiaire **(Ma.b)**. On peut aussi introduire des éléments plus complexes, en particulier les opérations sur **bits** comme par exemple une bascule **SR** (priorité déclenchement), **RS** (priorité enclenchement), **POS** et **NEG** pour la détection de fronts... on trouvera d'autres fonctions utiles, les compteurs, les temporisateurs et le registre à décalage.

On peut également utiliser des fonctions plus complexes (calculs sur mots par exemple).

7.3 Adressage des entrées/sorties :

La déclaration d'une entrée ou sortie donnée à l'intérieur d'un programme s'appelle l'adressage. Les entrées et sorties des API sont la plupart du temps regroupées en groupes de huit sur des modules d'entrées ou de sorties numériques. Cette unité de huit est appelée **octet**. Chaque groupe reçoit un numéro que l'on appelle l'**adresse d'octet**.

Afin de permettre l'adressage d'une entrée ou sortie à l'intérieur d'un octet, chaque octet est divisé en huit **bits**. Ces derniers sont numérotés de 0 à 7. On obtient ainsi l'**adresse du bit**. L'API représenté ici a les octets d'entrée 0 et 1 ainsi que les octets de sortie 0 et 1.

Nom	Type de données	Adresse	
ETAPE 1	Bool	M0.1	← Etape 1 de type logique (Bool) affecté à la mémoire M0.1
ETAPE 2	Bool	M0.2	
ETAPE 3	Bool	M0.3	
ETAPE 4	Bool	M0.4	
C1	Bool	I0.0	← Le capteur C1 est de type logique et affecté à l'adresse I0.0
C2	Bool	I0.1	
C3	Bool	I0.2	
C4	Bool	I0.3	
DCY	Bool	I0.4	
RO	Bool	Q0.0	
DE	Bool	Q0.1	← La sortie DE est de type logique et affecté à l'adresse Q0.0
MO	Bool	Q0.2	

Tableau (III.1) : table de variables mnémoniques.

Par exemple, pour adresser la 5ème entrée du **DCY** en partant de la gauche, on définit l'adresse suivante :

– I0.4 : **I** indique une adresse de type entrée, **0**, l'adresse d'octet et **4**, l'adresse de bit. Les adresses d'octet et de bit sont toujours séparées par un point.

Pour adresser la 3ème sortie, par exemple, on définit l'adresse suivante :

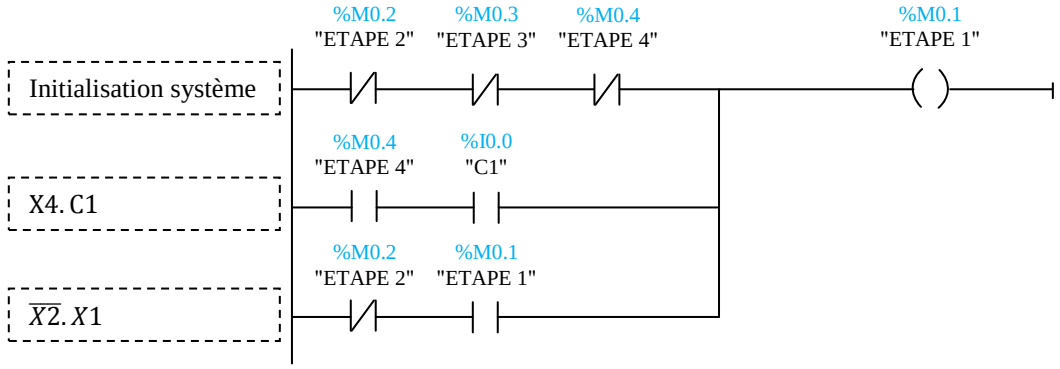
– Q0.2 : **Q** indique une adresse de type Sortie, **0**, l'adresse d'octet et **2**, l'adresse de bit. Les adresses d'octet et de bit sont toujours séparées par un point.

Remarque : L'adresse du bit de la dixième sortie est un 1 car la numérotation commence à zéro.

Exemple :

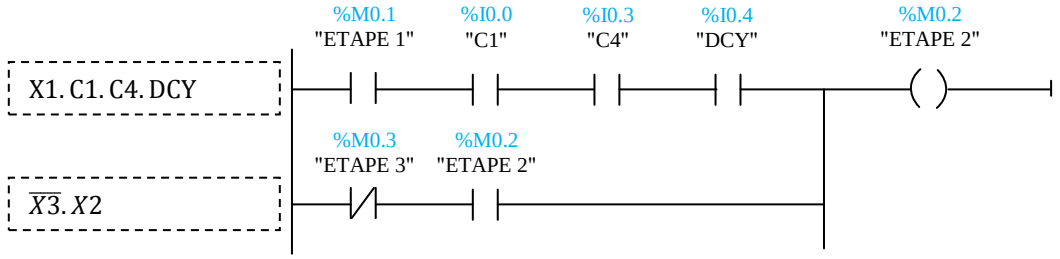
Dans l'exemple précédent et suivant la table mnémonique d'affectation le programme en LADER de la première étape est :

$X1 = \mathrm{X4.C1} + \overline{X2}.X1$



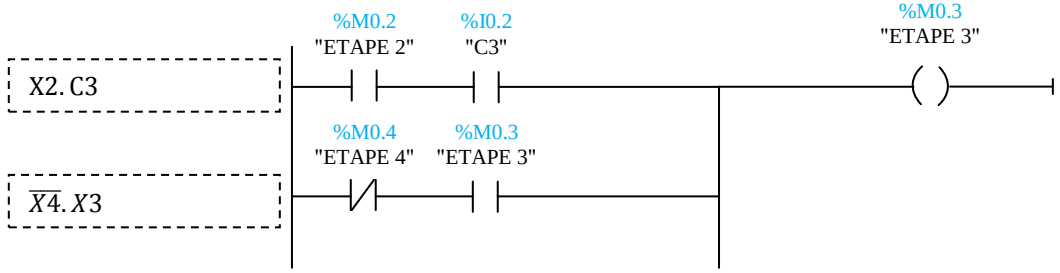
Et ainsi pour l'étape 2 est :

$X2 = \mathrm{X1.C1.C4.DCY} + \overline{X3}.X2$



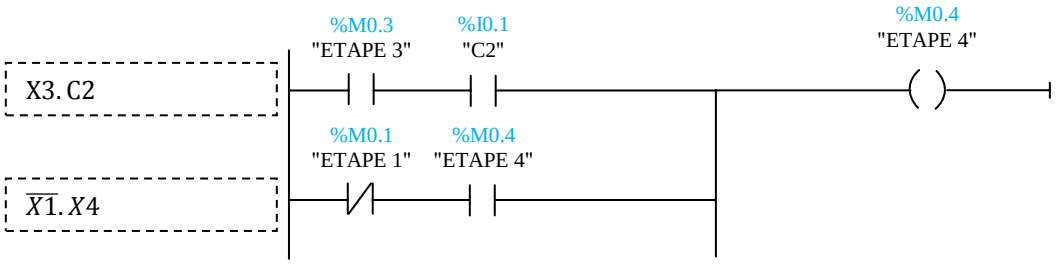
L'étape 3 :

$X3 = \mathrm{X2.C3} + \overline{X4}.X3$



L'étape 4 :

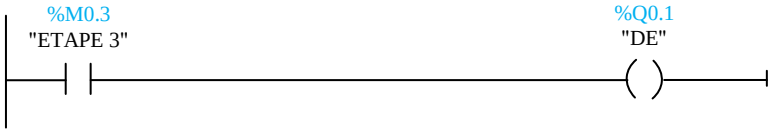
$X4 = \mathrm{X3.C2} + \overline{X1}.X4$



Pour la programmation des sorties :

Sortie RO : est actionné uniquement à l'étape 2

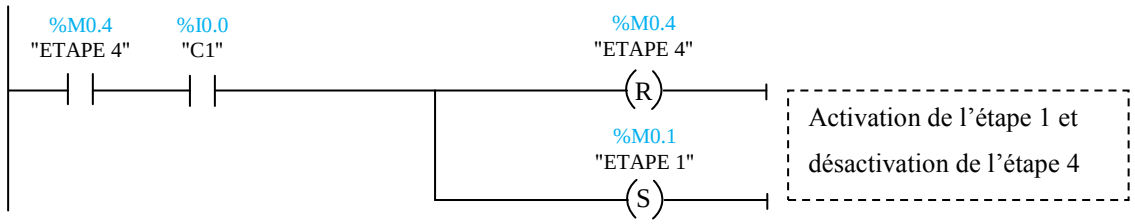
DE : est actionné uniquement à l'étape 3



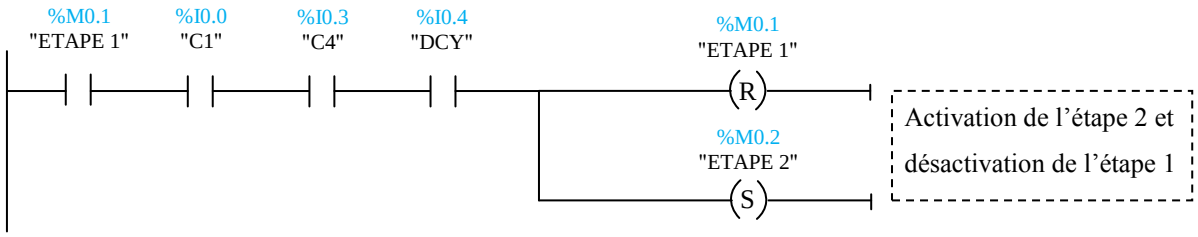
MO : est actionné uniquement à l'étape 4

Le programme peut être simplifié si en utilisant les bobines Set/ Reset ou les bascules SR ou RS et en tenant compte des cinq règles du GRAFCET

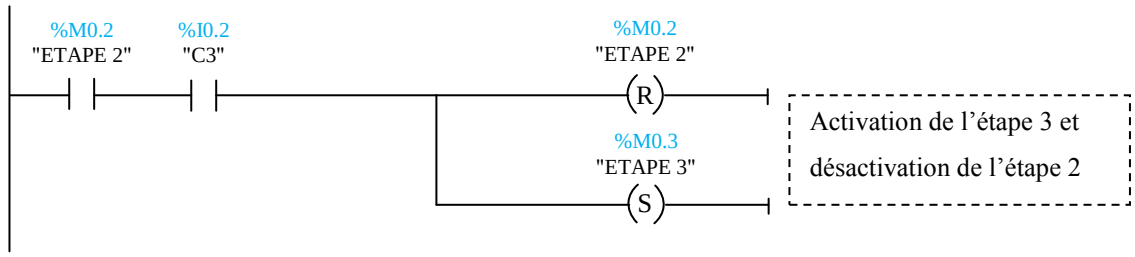
ETAPE 1



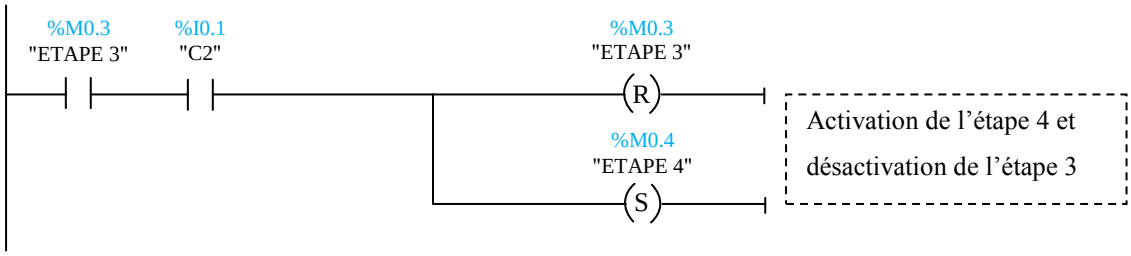
ETAPE 2



ETAPE 3



ETAPE 4



8. Le langage LOG (Logigramme) :

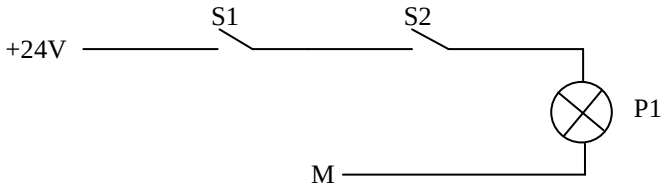
Les opérations logiques servent à définir des conditions pour l'activation d'une sortie. Elles peuvent être créées dans le programme de l'API dans les langages de programmation Schéma des circuits LADER (LD) ou Logigramme (LOG).

Il existe de nombreuses opérations logiques pouvant être mises en œuvre dans des programmes API.

L'opération ET et l'opération OU, ainsi que la NEGATION d'une entrée sont les opérations les plus fréquemment utilisées et seront expliquées ici à l'appui d'un exemple.

8.1 Opération ET :

Exemple : Une lampe doit s'allumer quand les deux interrupteurs sont fermés simultanément.



- Explication : La lampe s'allume uniquement quand les deux interrupteurs sont fermés. C'est-à-dire, quand S1 ET S2 sont fermés, alors la lampe P1 est allumée.
- Câblage de l'API : Pour appliquer cette opération au programme de l'API, les deux commutateurs doivent être connectés aux entrées de l'API. Ici, S1 est relié à l'entrée I 0.0 et S2 à l'entrée I 0.1. De plus, la lampe P1 doit être connectée à une sortie, par exemple Q 0.0.

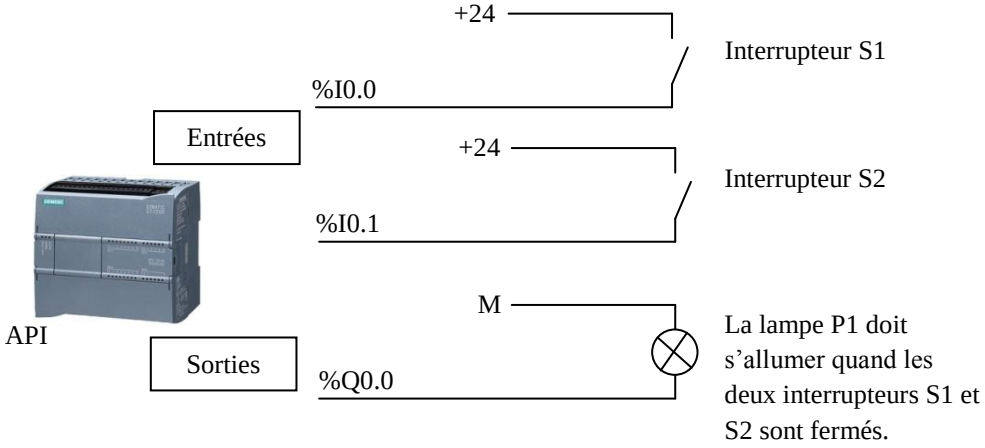
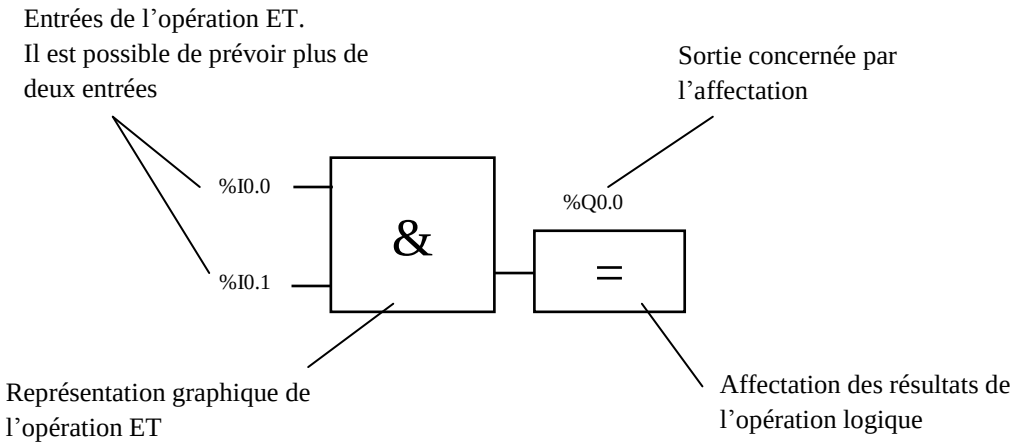


Figure (III.12) : Opération ET.

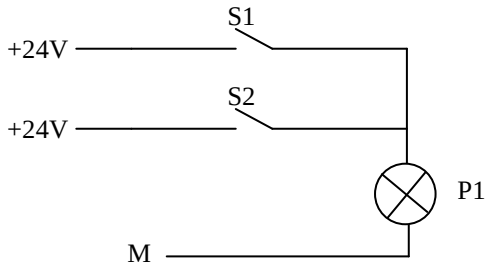
8.2 Opération ET dans LOG :

Dans le logigramme LOG, l'opérateur ET est programmé par le symbole ci-dessous et est représenté de la manière suivante :



8.3 Opération OU :

Exemple : Une lampe doit s'allumer si au moins un des deux interrupteurs est fermé.



- Explication : La lampe s'allume à partir du moment où un des deux interrupteurs est fermé. C'est-à-dire, quand S1 OU S2 est fermé, alors la lampe P1 est allumée.
- Câblage de l'API : Pour appliquer cette opération au programme de l'API, les deux commutateurs doivent être connectés aux entrées de l'API. Ici, S1 est relié à l'entrée E 0.0 et S2 à l'entrée E 0.1. De plus, la lampe P1 doit être connectée à une sortie, par exemple A 0.0.

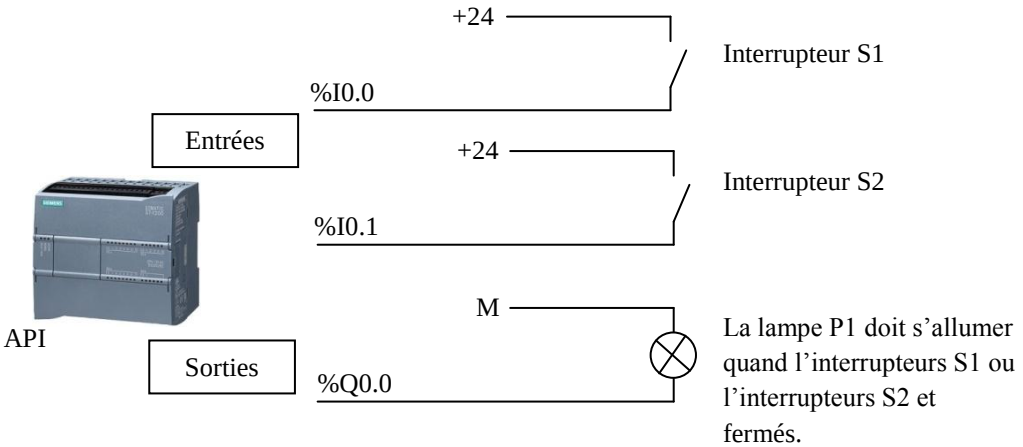
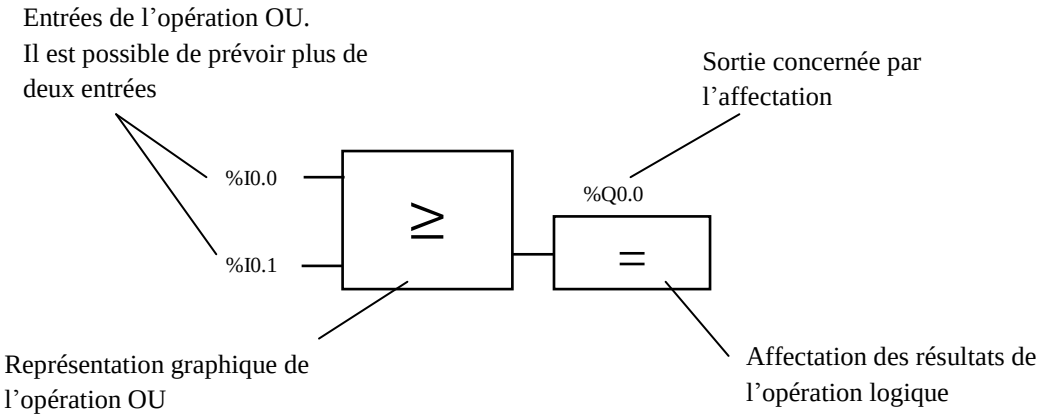


Figure (III.13) : Opération OU.

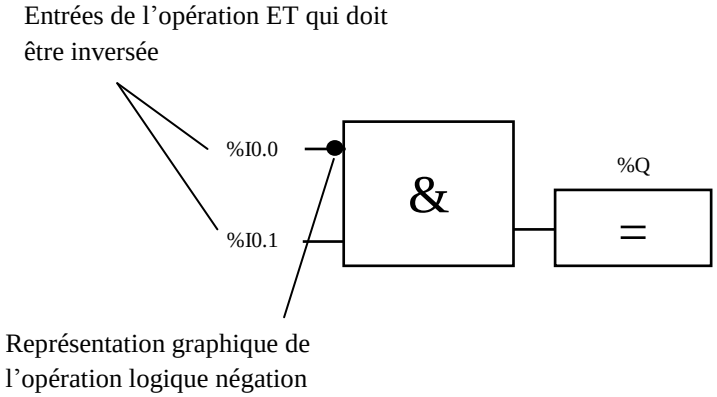
8.4 Opérateur OU dans LOG :

Dans le logigramme LOG, l'opérateur OU est programmé par le symbole ci-dessous et est représenté de la manière suivante :



8.5 Négation :

Il est souvent nécessaire dans les opérations logiques d'interroger l'état d'un contact pour savoir : dans le cas d'un contact à fermeture si celui-ci n'a pas été activé, ou dans le cas d'un contact à ouverture s'il a été activé, et donc pour savoir si la tension est appliquée à la sortie ou non. Ceci peut être réalisé par la programmation d'une négation à l'entrée de l'opération ET ou OU. Dans le logigramme LOG, la négation de l'entrée (ou inversion) sur un opérateur ET est programmé de la façon suivante :



Ceci signifie qu'une tension est appliquée à la sortie %Q 0.0 uniquement si %I 0.0 est à 0 et %I 0.1 est à 1.

9. Conclusion :

Ces dernières années, les avancées technologiques ont conduit au développement des automates programmables industriels (API) et à une révolution importante dans l'automatique.

Ce chapitre constitue une introduction aux automates programmables industriels (API), ainsi qu'à leur fonction générale, leurs formes matérielles et leur architecture interne. Les API sont employés dans de nombreuses tâches d'automatisation, dans différents domaines, comme les processus de fabrication industriels.